Designing a MySQL Database: Tips and Techniques

(2009-07-13) - Contributed by Codex-M

You are planning to develop a PHP web application that needs a MySQL database. Now what? You need to design your MySQL database first. Need a little help? Keep reading.

Most developers do not have a design background regarding MySQL, so when they create their database, it's inefficient, buggy and unreliable.

By properly setting up the correct specifications for your MySQL database, your web application will be efficient and reliable, since it meets industry standards for properly setting up a database.

You gain the following benefits from properly designing a MySQL database:

- Speed - Because the database is efficient and uses the correct data types and character length, query speed will improve.

- Security - Security improves because you allow specific data to be stored in the MySQL database, and nothing else.

- Storage space efficiency - Since you are developing a database that is purely based on data specifications, there will be no unnecessary bytes or characters stored in the MySQL database. This saves disk space.

This tutorial will focus on designing a MySQL database for PHP web applications and using the phpMyAdmin interface. Below are the requirements for you to design your database (starting from scratch):

- XAMPP, for testing PHP scripts and accessing phpMyAdmin. If you're not familiar with it, read this XAMPP article for more information.

- A List of fields and their data types. You'll need a complete understanding of the data to be processed and stored.

- The number of characters in every piece of data to be stored.

Please note that this tutorial has been tested under MySQL version 5.0.67. Different MySQL versions might affect the design of your database, but basically the concept will be the same. This tutorial also assumes that the table format to be used will be MyISAM, which is the default table for MySQL.

{mospagebreak title=Naming the Database, Tables, Field names and Data Types}

To help you easily understand the processes, we will present the tutorial with a real life implementation. Let us assume you are given a task to design a web application that will receive customer complaints in your website. During your brainstorming session, you came up with the following data that need to be gathered from the web form every time there is a customer-related complaint:

- Customer name (Example: John Doe).

- Price (Example: 34878.456). This should not exceed 4 digits before the decimal point and allow only 3 digits after the decimal point. Also, the stored values should not be negative.

- Date purchased (using this format: 2009-05-16, or yyyy-mm-dd).

- Product Serial Key (integer only, should not be negative values, up to 5 digits, use zeros to fill in if less than 5 digits) Example: 00567, 56453, 00021.

- Complaint details (accept text input of any length).

- Receiving date of complaint (displays the current time and date the form was submitted).

Now that we have clearly defined the data that needs to be gathered, the next step is to assign a name to the database and the table.

According to the MySQL guidelines, the database name, table name and field names should not exceed 64 characters in length, and ideally should not contain special characters, to avoid the use of a back tick.

For simplicity we can name the database "customercomplaint," and then we can name the database table "customertable." The more you can simplify the naming, the more it will be convenient to use it in the associated PHP scripts. Avoid using difficult or mixed-case database names, tables and field names because it can create confusion when you formulate the PHP queries to the database.

For the corresponding data types for each of those variables, below is a screen shot of the available data types for MySQL:

The most commonly used MySQL data types are VARCHAR, DECIMAL, DATE, INT, TEXT and TIMESTAMP.

VARCHAR is commonly used for variable length strings up to 255 characters. If you are storing and processing data that is alphanumeric in nature, this data type is the most suitable. Common examples of actual data that fits this type include people's names, zip codes, telephone numbers and any type of alphanumeric data not exceeding 255 characters in length. Do not use the VARCHAR type when storing numbers which will be used for computation; it might cause some computation-related problems. In other words, it might affect the accuracy and integrity of the computation.

DECIMAL is appropriate for storing numbers that will be used for computation. In MySQL, we can specify the number of allowable digits in numbers to be stored (including the number of decimal places). We can also specify whether negative values are allowed.

Assigning the proper length of the DECIMAL type can be tricky. For example, if you need to store only 5 digits before the decimal point and allow only 3 digits after the decimal point, the proper length/values to be configured in the database will be:

Decimal (5+3, 3) or Decimal (8, 3)

Examples of allowed numbers include: 12345.678, 56872.690, 11.6 and 12.568

The following numbers, however, will return an error: 128781.1, 8972865.231

The DATE data type is recommended for storing dates. The default date format for MySQL is 2009-05-18 [year-month-day].

The INT data type is recommended if you are storing numbers which will not contain a decimal point. INT stands for integer. Again, like DECIMAL, correctly specifying INT in MySQL could be tricky.

There are several integer types which you need to know, as well as the maximum number of digits they can have:

- TINYINT - This type will accept up to 3 digits as the maximum.

- SMALLINT- Allows up to 5 digits maximum.

- MEDIUMINT - This type will accept up to 8 digits as the maximum.

- INT - This type will let you go up to 10 digits.

- BIGINT - If you plan to allow up to 20 digits, this is recommended.

TEXT is a very useful data type that will accept text inputs, a mixture of just any characters that comprise the content of any web form inquiry. VARCHAR can only accept up to 255 characters, but TEXT can be used to store data that exceeds that amount.

When the TIMESTAMP data type is selected, by checking "CURRENT_TIMESTAMP" as the default, MySQL automatically returns the actual date and time of every MySQL data insertion.

{mospagebreak title=Finalization of Database Design Specifications}

Now that we have basic knowledge of different MySQL data types and rules in naming the database and tables, we can finalize the database specifications of your web application. It is highly recommended that you write down the specifications on a piece of paper or in a Microsoft Word file regarding your planned structure of the database before submitting a query to MySQL.

Below are the recommended specifications:

First field:

Data to be gathered: Customer name

Recommended Field name: customername

MySQL data type: VARCHAR

Maximum length of customer name allowed: 64 (this can be extended, for illustration purposes only)

Second field:

Data to be gathered: Price

Recommended Field name: price

MySQL data type: DECIMAL

Maximum number of allowable digits before decimal point: 4

Maximum number of decimal places: 3

Final decimal length specification: DECIMAL (4+3, 3) OR

DECIMAL (7, 3)

No negative values allowed

Third field:

Data to be gathered: Date purchased

Recommended Field name: datepurchased

MySQL data type: DATE

Maximum Length: not applicable

Fourth field:

Data to be gathered: Product Serial Key

Recommended Field name: productkey

MySQL data type: SMALLINT

Maximum character length of this variable: 5

Other attributes: Zero-fill if less than 5 digits; no negative values allowed

Fifth field:

Data to be gathered: Complaint details

Recommended Field name: complaintdetails

MySQL data type: TEXT

Maximum Character length: Depends on user input

Sixth field:

Data to be gathered: Receiving date of complaint

Recommended Field name: receivingdate

MySQL data type: TIMESTAMP

Other attributes: Set "default" value to "Current_Timestamp" so that it will return the actual date of every MySQL data insertion.

{mospagebreak title=Inputting the specifications into MySQL using phpMyAdmin}

We will now input these specifications into the MySQL environment using the phpMyAdmin graphical interface. Follow these short steps:

Step 1: Log in to phpMyAdmin using your MySQL username and password.

Step 2: Click the "database" link, and in the "Create new database" section enter customercomplaint.

Step 3: On the "Create new table on database customercomplaint," section, enter "customertable" and in the number of fields, enter 6. Click "go."

Use the design specifications to configure the variables in the database. See the screen shot below for a complete guide:

Do not forget to check "Current_Timestamp" under the Default column in receivingdate field. "Unsigned" means it will accept only positive values, which can be set under "Attributes". "Zero Fill" means it will insert zeroes for spaces if the number of digits entered is less than five.

For example, if the product serial key is 456, MySQL will store it as "00456."

The detailed MySQL query for creating the designed table is as follows:

```
CREATE TABLE `customercomplaint`.`customertable` (

`customername` VARCHAR( 64 ) NOT NULL ,`price` DECIMAL( 7, 3 ) UNSIGNED NOT NULL ,`datepurchased` DATE NOT NULL ,`productkey` SMALLINT( 5 ) UNSIGNED ZEROFILL NOT NULL ,`complaintdetails` TEXT NOT NULL ,`receivingdate` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP

) ENGINE = MYISAM
```

After everything is set, click "Save." You can now start inserting data into your newly-designed database. You will notice it will only accept data according to your specifications and display errors for those are not within the design.